ISSN: 2586-7652 (Print) Vol. 03, No. 02, September 2020
ISSN: 2635-7607 (Online)
International Journal of Advanced Engineering
Source: http://ictaes.org
Manuscript received: July 30,2020 ; Revised: September 25, 2020 ; Accepted: September 29, 2020

Optimizing Tremaux Algorithm in Micromouse Using Potential Values

Sanjay Rijal [1], Rabin Nepal [2], Rhimesh Lwagun [3], Rohit Pati [4], Janardan Bhatta [5]

[1, 2, 3, 4, 5] *Institute of Engineering Thapathali Campus, Tribhuvan University, Thapathali, Kathmandu, Nepal*

zovelsanj28@gmail.com[1], rabin47nepal@gmail.com[2], rimeshlwagun10@gmail.com[3], patirohit26@gmail.com[4], Janardan.bhatta@ioe.edu.np[5]

Abstract

This paper discusses the furtherance of the Tremaux algorithm by employing a potential value algorithm in conjunction, for improving search in a micromouse. The fused algorithm running on the STM32 Bluepill microcontroller explores and finds the shortest path in a 16x16 maze. Proximity sensors, gyroscope, and magnetometer together with encoder motors aid the micromouse to understand its surrounding hindrances and make precise movements while traversing through a maze. The optimized algorithm eliminates any paths that may lead the micromouse further away from the center of the maze during the initial run itself and saves a significant amount of time while solving a maze.

Keywords: *Tremaux algorithm, potential value, micromouse, STM32 Bluepill, 16x16 maze*

1. Introduction

Automated systems possess vast scopes in autonomous locomotion. Tasks such as obstacle detection and maze solving are some of the challenges to an automatic vehicle. The objective of this paper is to present an algorithm for real-time maze solving and shortest path determination. Designing an algorithm for any maze is efficient if observed from the graph theory perspective. Graphs are discrete structures consisting of vertices and edges that connect these vertices [1]. A maze can be viewed as an undirected disconnected non-simple graph. The maze may be of different dimensions 8x8, 16x16, 32x32, and so on. Each cell can be considered as a node isolated by walls (edges). Through proper mathematical modeling and interpretation of graph theory into a maze, it is possible to find the shortest path between any two nodes [2]. Till date, various maze-solving algorithms have been designed and implemented such as wall follower algorithm, pledge algorithm, flood-fill algorithm, dead-end filling algorithm, Tremaux algorithm, and so on [3]. The selection of algorithms depends on various factors

Corresponding Author: Sanjay Rijal
Author's affiliation: Institute of Engineering Thapathali Campus, Tribhuvan University
Email: zovelsanj28@gmail.com
Copyright © ICT-AES

such as rules for maze solving, the complexity of mazes, time limitation, hardware constraints, etc. The algorithm design explained in this paper was originally designed for International Micromouse Competition at Techfest 2019/2020, IIT Bombay, based on rules standardized by IEEE.

2. Design of Searching Algorithm

2.1. Elementary Algorithm

The Tremaux algorithm is based on the Tremaux tree which was named after Charles Pierre Tremaux, a 19th-century French author who used a form of depth-first search as a strategy for solving mazes. In graph theory, the Tremaux tree of an undirected graph G is a spanning tree of G, rooted at one of its vertices with the property that every two adjacent vertices in G are related to each other as an ancestral and descendant in the tree [4]. It uses a marking procedure for maze exploration. The nodes or cells are marked once, twice, or left unmarked. The nodes left unmarked are the unvisited nodes while those marked once are visited once. The nodes marked twice are of greater concern. Each of the nodes marked twice will be bidirectional: once forward and once the other way around, and often leading to dead-ends. These nodes will be treated as dead-ends and thus not considered in the final optimal path. As a result, the nodes visited twice, loops, and dead-ends are eliminated from the optimal path. If a path is found, the nodes marked only once indicate a path that connects entrance and destination, thus finding only one solution [5].

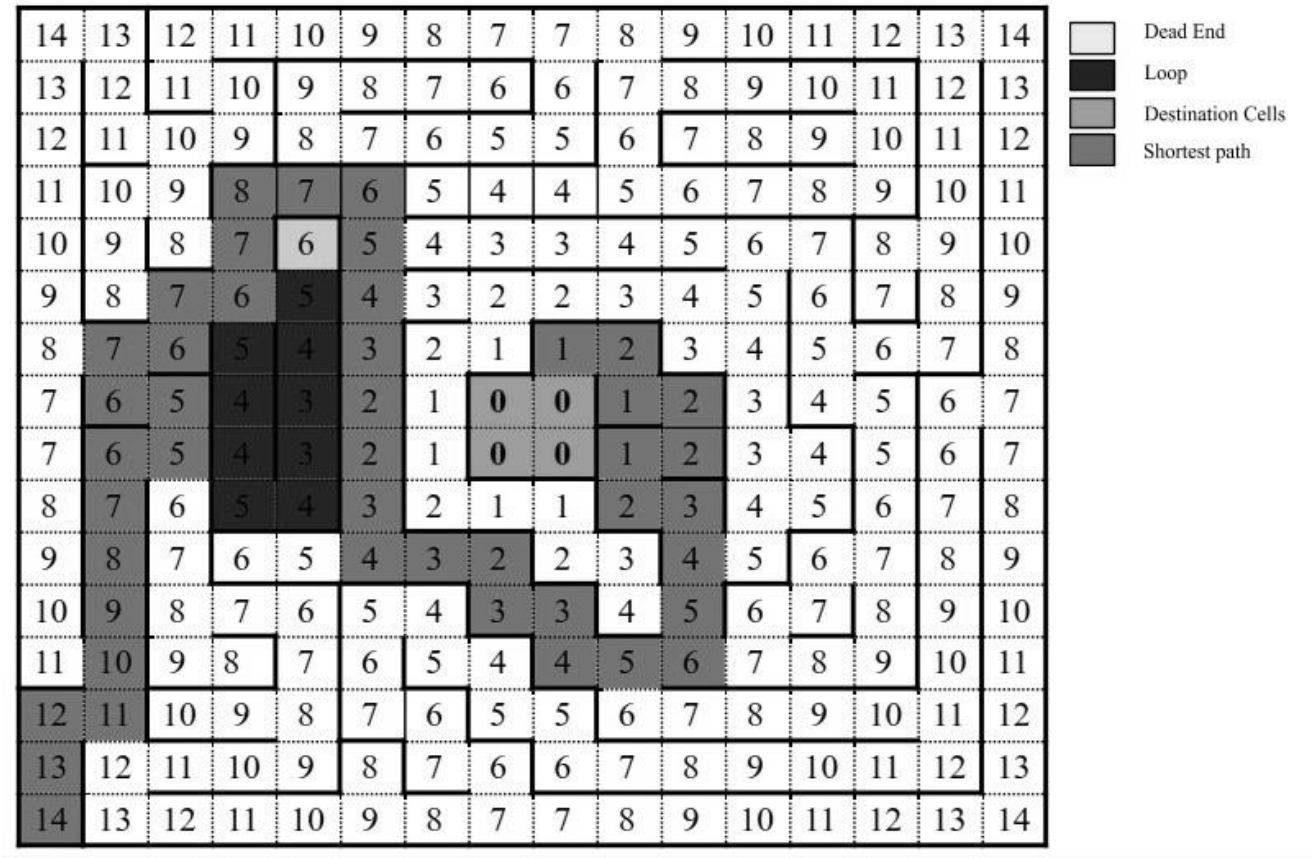


Figure 1. Potential Value for a 16x16 maze, dark lines represent walls whereas dashed lines represent the absence of walls

The Tremaux algorithm is guaranteed to work for any maze with well-defined paths but its shortest path may not be the quickest one. To solve this problem the potential-value algorithm comes into action. The potential

value is any non-negative value possessed by any cell representing the minimum distance in numbered rectangular steps from the destination cells located at the center of the maze [6]. The destination cells have zero values (0) and those of other cells increase by one as their distance from the center cells increase. The potential value of each cell for a 16x16 maze is shown in Figure 1.

2.2. Proposed Algorithm

The proposed searching algorithm is the combination of both the Tremaux and potential-value algorithm. The marking procedures are the same as that of the Tremaux algorithm while potential values are given priority over the markings to direct the micromouse towards the destination which gives an optimal path quicker than that obtained from the Tremaux algorithm.

Heading defines the direction to which the front face of the micromouse is facing i.e. East, West, North, and South. The starting position of the robot is from one of the four corners. As the maze solving algorithm is called, the robot at its starting position functions as shown by the flowchart in Figure 3. Initially, all the cells are marked unvisited and the potential values are mapped. Then it checks the presence of walls and determines how many paths are possible from the standing node or cell. The decisions are made in the following order:

1. Call for a dead-end (possible paths = 0) if walls are found on all three sides. The robot turns 180 degrees and returns back to the previous cell.
2. Call for a 1-way (possible paths = 1) path if the wall is absent on only one side. The robot then calls the Tremaux part of the algorithm and checks if the cell next to it is marked twice. If the cell is marked twice it is treated as a dead-end else it marks the present standing cell and moves to the next one.
3. Execute a 2-way (possible paths = 2) path if the wall is not found on two sides. The robot then compares the two cells and calls the potential-value part of the algorithm. If the cell with a smaller value is unmarked then the robot visits the cell after marking the present cell, else the cell with a higher value is visited. If both the cells have the same potential value and are unmarked then any one of them can be chosen. If the cells are marked more than once then it is considered a dead-end.
4. Execute a 3-way (possible paths = 3) path if no walls are detected on any of the sides. The robot then compares the potential values of all three cells, marks the standing cell, and visits the cell with the lowest value. If any of the two cells have the same lower value, then anyone of them can be chosen. If a cell is marked twice then it is considered as a loop and the robot chooses other paths.

The shortest path thus will be the cells joining the starting and the destination cells marked only once. It excludes dead-ends, loops and the cells visited more than once. The shortest path given by the proposed algorithm is as shown in Figure 1.

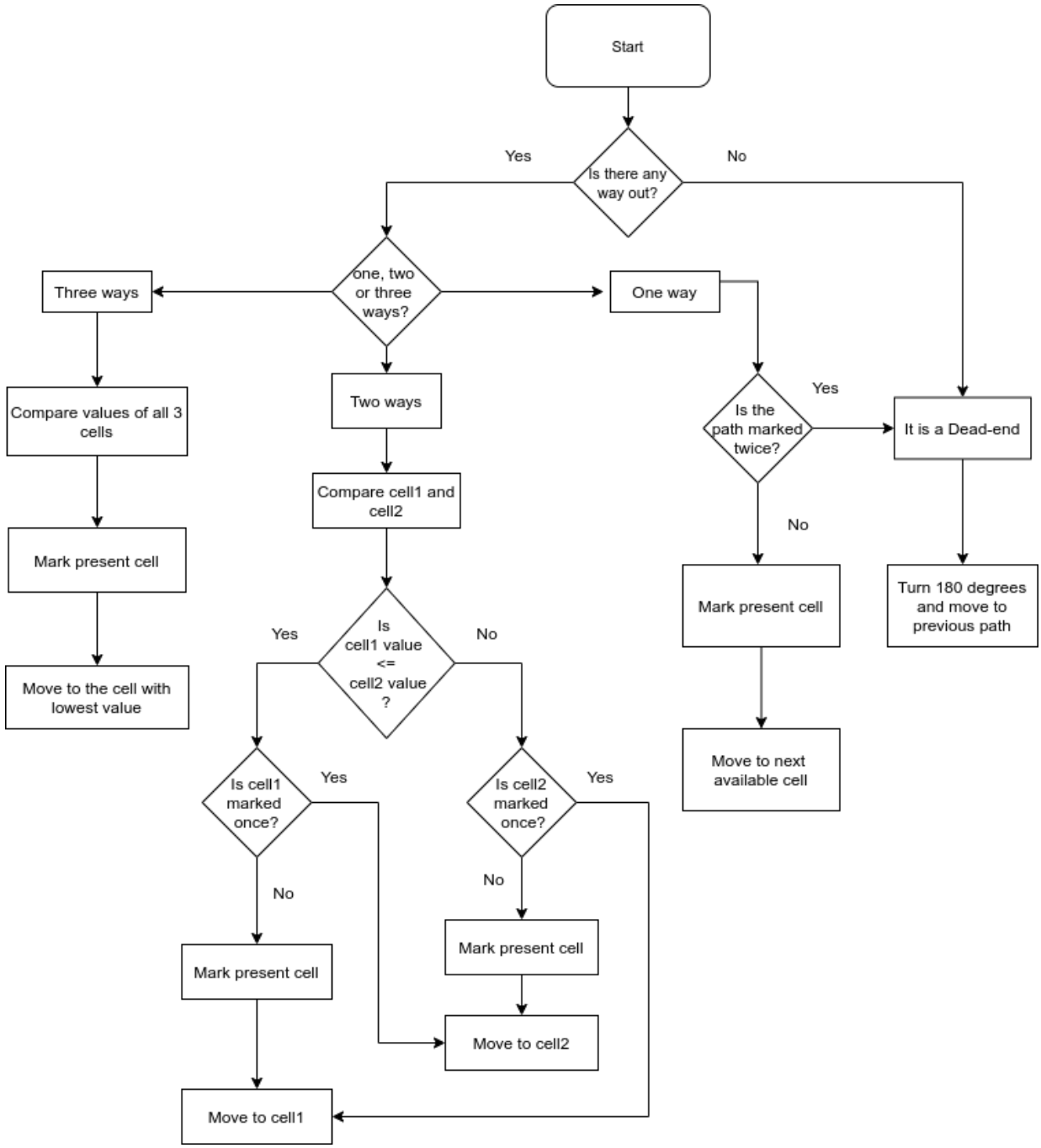


Figure 2. Flow-chart of Proposed Algorithm

3. Result

The proposed algorithm was tested with five different types of mazes of complexities ranging from easy to hard in a micromouse simulator program called mms developed in C++ programming language [7] and in a real maze as well. Tremaux and the proposed algorithm were evaluated in mms and the efficiency measure was based on the quickest path rather than the shortest path. The time comparison between the Tremaux algorithm with and without potential-value algorithm for different mazes are as shown in Table 1.

Table 1. Evaluation of Tremaux and Proposed Algorithm

Tremaux Algorithm		Proposed Algorithm	
Search Time (in sec)	Run Time (in sec)	Search Time (in sec)	Run Time (in sec)
234	96	241	96
313	52	34	34
379	101	112	88
376	58	295	58
345	102	351	102

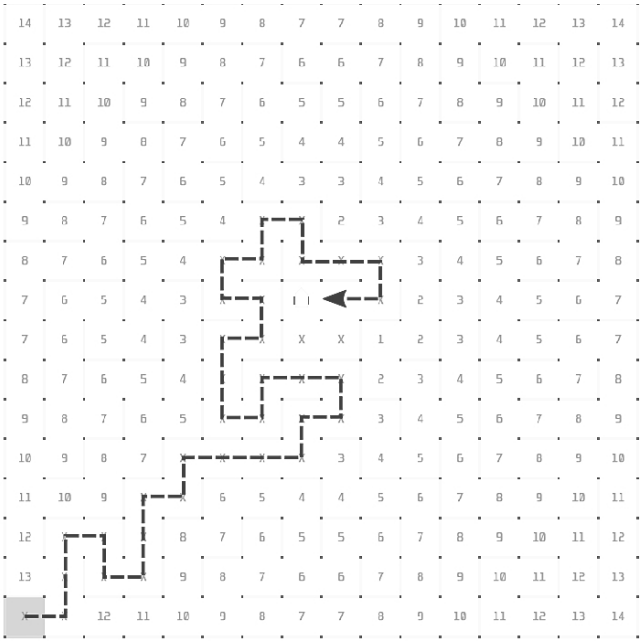


Figure 3. Shortest Path from Tremaux Algorithm

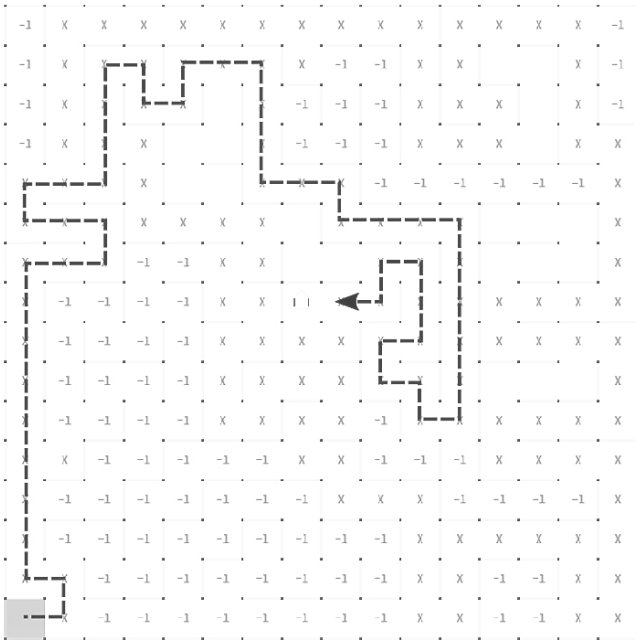


Figure 4. Shortest Path from Proposed Algorithm

In a real maze, the more efficient algorithm between the two was implemented as the final algorithm for International Micromouse Competition. For search time, the rpm of the motor was roughly 108 and the total search time was 3 minutes 41 seconds. For run time, the rpm of the motor was doubled and the total run time was 1 minute 14 seconds.

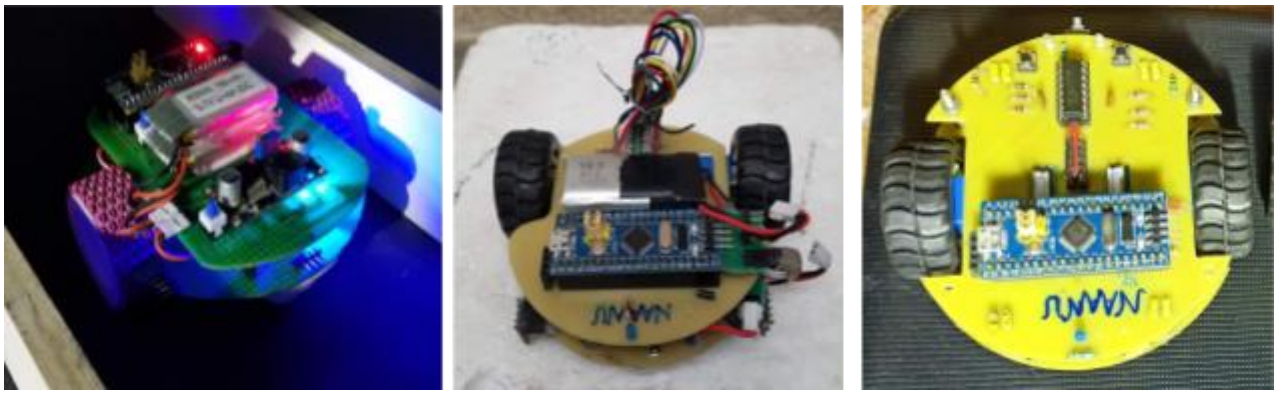


Figure 3. Different Versions of Fabricated Micromouse

4. Analysis and Discussion

Through the analysis of the performance of the micromouse in simulation as well as in a real maze, the average improvements in search time and run time of the proposed algorithm were obtained as 35.28% and 9.50% respectively compared to that of the bare Tremaux algorithm. The proposed algorithm was able to direct the micromouse further towards the center saving time on unnecessary exploration. Exploiting very few variables and loops, the proposed algorithm consumed less memory space. Along with these promising results, some of its demerits are; it works only when the robot has prior knowledge of the destination cells, and following the lower potentials may not always lead the quickest path.

During implementation in a real maze, the frictional and inertial factors along with hardware constraints showed a direct impact on the performance of the micromouse. The hardware can be refined and the effects of friction and inertia can be compensated in hardware as well as in software. Further enhancement in the performance can be through the implementation of smooth turnings instead of stop-and-run motion.

5. Conclusion

The proposed algorithm addresses the limitation of one algorithm with the help of another. The Tremaux algorithm can solve any maze but shows poor performance in exploration and execution with respect to time whereas potential value cannot keep track of the traversed paths. Employing an ensembled algorithm that exploits both Tremaux and potential value algorithms improves the searching process in a micromouse remarkably.

Acknowledgement

The research was supported by Robotics and Automation Center, Institute Of Engineering Thapathali Campus both technically and financially during all phases of the work.

References

[1] Rosen, K. H. (2012). *Discrete Mathematics and Applications* (7th ed.). McGraw-Hill.

[2] Adil M. j. Sadik. (2010). A Comprehensive and Comparative Study of Maze-Solving Techniques by Implementing Graph Theory. *International Conference on Artificial Intelligence and Computational Intelligence.*

[3] L. Bienias, K. Szczepanski, P. Duch. (2016). Maze Exploration Algorithm for Small Mobile Platforms. *Image Processing & Communications, 21*, 15-26.

[4] Even, S. (2011). *Graph Algorithms.* Cambridge University Press.

[5] Lim Kai Li, Fawaz Y. Annaz. (2014). Implementation of the Trémaux Maze Solving Algorithm to an Omnidirectional Mobile Robot. *ICEIC.* Kota Kinabalu, Malaysia.

[6] B. Arthaya, A. Sadiyoko, A. Hadiwidjaja. (2006). The Design of a Maze Solving System for a Micromouse by using a Potential Value. *World Transactions on Engineering and Technology Education, 5*(3).

[7] Macorone, M. (n.d.). Retrieved from https://github.com/mackorone/mms